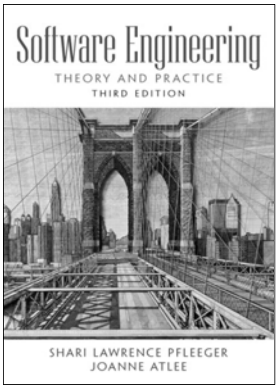
Chapter 12

Evaluating Products, Processes, and Resources



ISBN 0-13-146913-4
Prentice-Hall, 2006

Contents

Chapter 12 Objectives

- Feature analysis, case studies, surveys, and experiments
- Measurement and validation
- Capability maturity, ISO 9000, and other process models
- People maturity
- Evaluating development artifacts
- Return of investment

12.1 Approaches to Evaluation

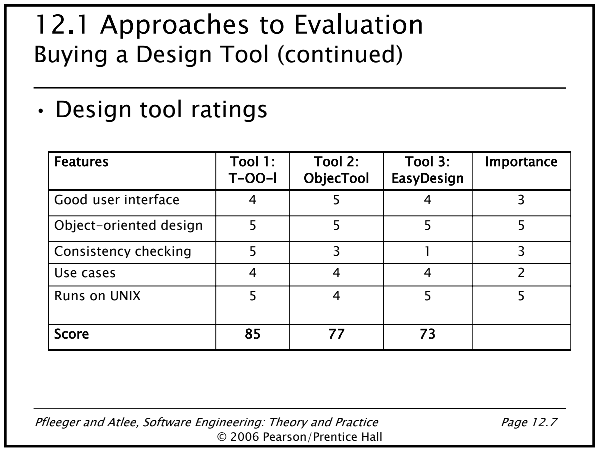
- Measure key aspects of product, processes, and resources
- Determine whether we have met goals for productivity, performance, quality, and other desire attributes

12.1 Approaches to Evaluation
Categories of Evaluation

- **Feature analysis**: rate and rank attributes
- **Survey**: document relationships
- Case studies
- Formal experiment

12.1 Approaches to Evaluation
Feature Analysis Example: Buying a Design Tool

- List five key attributes that the tool should have
- Identify three possible tools and rate the criterion
- Examine the scores, creating a total score based on the importance of each criterion
- Based on the score, we select the highest score (t-OO-1)

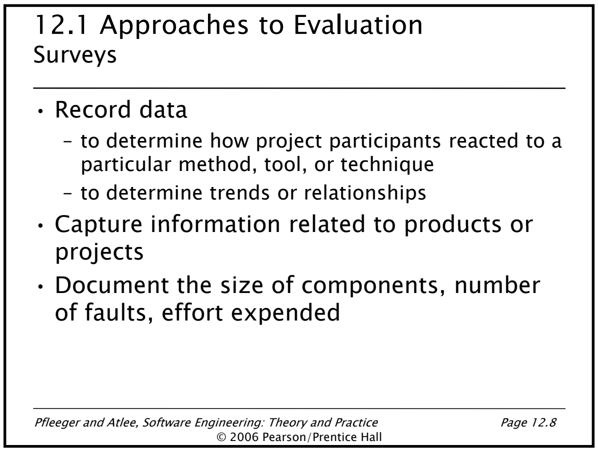


12.1 Approaches to Evaluation
Buying a Design Tool (continued)

- Design tool ratings

Features	Tool 1: T-OO-l	Tool 2: ObjecTool	Tool 3: EasyDesign	Importance
Good user interface	4	5	4	3
Object-oriented design	5	5	5	5
Consistency checking	5	3	1	3
Use cases	4	4	4	2
Runs on UNIX	5	4	5	5
Score	**85**	**77**	**73**	

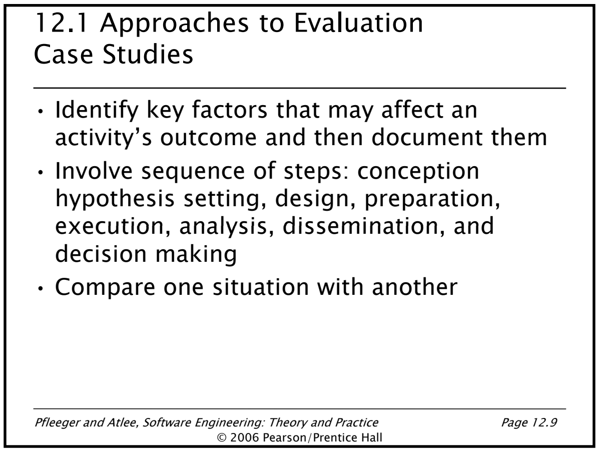
Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.7*
© 2006 Pearson/Prentice Hall



12.1 Approaches to Evaluation
Surveys

- Record data
  - to determine how project participants reacted to a particular method, tool, or technique
  - to determine trends or relationships
- Capture information related to products or projects
- Document the size of components, number of faults, effort expended

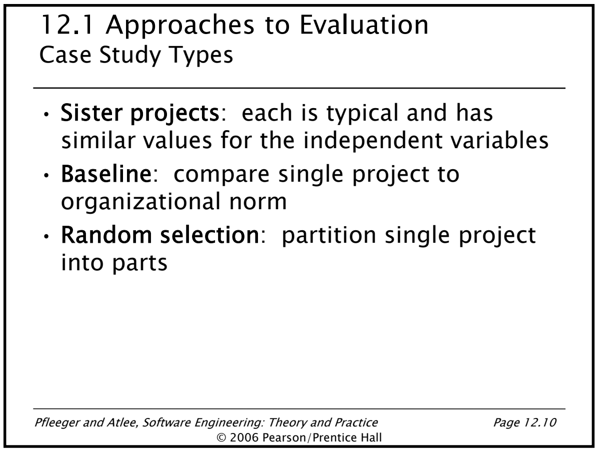
Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.8*
© 2006 Pearson/Prentice Hall



12.1 Approaches to Evaluation
Case Studies

- Identify key factors that may affect an activity's outcome and then document them
- Involve sequence of steps: conception hypothesis setting, design, preparation, execution, analysis, dissemination, and decision making
- Compare one situation with another

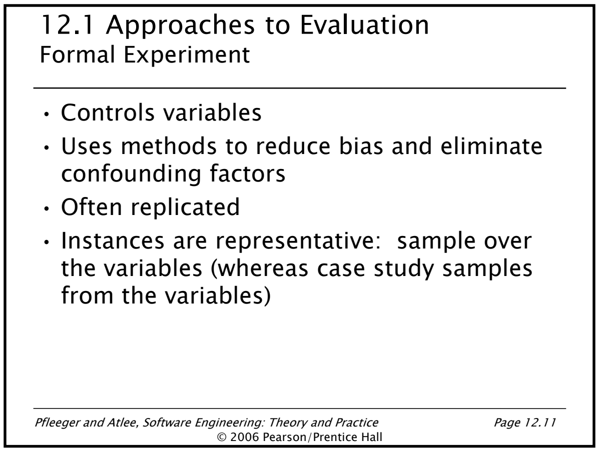
Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.9*
© 2006 Pearson/Prentice Hall



12.1 Approaches to Evaluation
Case Study Types

- **Sister projects**: each is typical and has similar values for the independent variables
- **Baseline**: compare single project to organizational norm
- **Random selection**: partition single project into parts

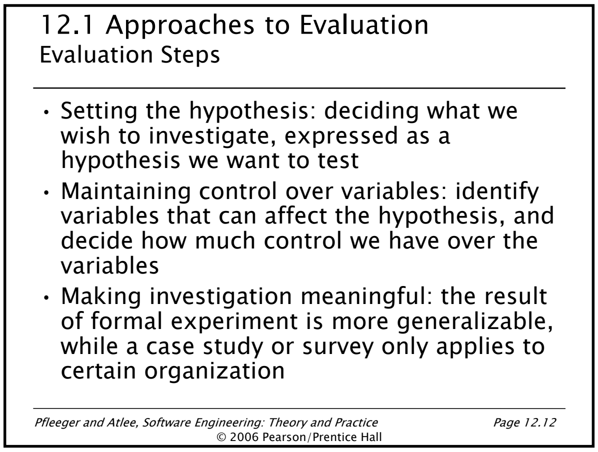
Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.10*
© 2006 Pearson/Prentice Hall



12.1 Approaches to Evaluation
Formal Experiment

- Controls variables
- Uses methods to reduce bias and eliminate confounding factors
- Often replicated
- Instances are representative: sample over the variables (whereas case study samples from the variables)

Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.11*
© 2006 Pearson/Prentice Hall



12.1 Approaches to Evaluation
Evaluation Steps

- Setting the hypothesis: deciding what we wish to investigate, expressed as a hypothesis we want to test
- Maintaining control over variables: identify variables that can affect the hypothesis, and decide how much control we have over the variables
- Making investigation meaningful: the result of formal experiment is more generalizable, while a case study or survey only applies to certain organization

Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.12*
© 2006 Pearson/Prentice Hall

12.2 Selecting An Evaluation Technique

- Formal experiments: research in the small
- Case studies: research in the typical
- Surveys: research in the large

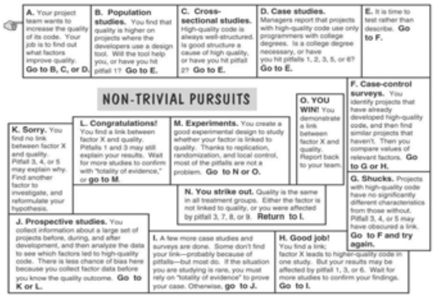
12.2 Selecting An Evaluation Technique
Key Selection Factors

- Level of control over the variables
- Degree to which the task can be isolated from the rest of the development process
- Degree to which we can replicate the basic situation

12.2 Selecting An Evaluation Technique
What to Believe

- When results conflict, how do we know which study to believe?
  - Using series of questions, represented by the game board
- How do you know if the result is valid?
  - Evaluation pitfalls table

12.2 Selecting An Evaluation Technique
Investigation and Evaluation Board Game



NON-TRIVIAL PURSUITS

A. Your project team wants to increase the quality of its code. Your job is to find out what factors improve quality. **Go to B, C, or D.**

B. Population studies. You find that quality is higher on projects where the developers use a design tool. Will the tool help you, or have you hit pitfall 1? **Go to E.**

C. Cross-sectional studies. High-quality code is always well-structured. Is good structure a cause of high quality, or have you hit pitfall 2? **Go to E.**

D. Case studies. Managers report that projects with high-quality code use only programmers with college degrees. Is a college degree necessary, or have you hit pitfalls 1, 2, 3, 5, or 6? **Go to E.**

E. It is time to test rather than describe. **Go to F.**

F. Case-control surveys. You identify projects that have already developed high-quality code, and then find similar projects that haven't. Then you compare values of relevant factors. **Go to G or H.**

G. Shucks. Projects with high-quality code have no significantly different characteristics from those without. Pitfall 3, 4, or 5 may have obscured a link. **Go to F and try again.**

H. Good job! You find a link; factor X leads to higher-quality code in one study. But your results may be affected by pitfall 1, 3, or 6. Wait for more studies to confirm your findings. **Go to I.**

I. A few more case studies and surveys are done. Some don't find your link—probably because of pitfalls—but most do. If the situation you are studying is rare, you must rely on "totality of evidence" to prove your case. Otherwise, **go to J.**

J. Prospective studies. You collect information about a large set of projects before, during, and after development, and then analyze the data to see which factors led to high-quality code. There is less chance of bias here because you collect factor data before you know the quality outcome. **Go to K or L.**

K. Sorry. You find no link between factor X and quality. Pitfall 3, 4, or 5 may explain why. Find another factor to investigate, and reformulate your hypothesis.

L. Congratulations! You find a link between factor X and quality. Pitfalls 1 and 3 may still explain your results. Wait for more studies to confirm with "totality of evidence," or **go to M.**

M. Experiments. You create a good experimental design to study whether your factor is linked to quality. Thanks to replication, randomization, and local control, most of the pitfalls are not a problem. **Go to N or O.**

N. You strike out. Quality is the same in all treatment groups. Either the factor is not linked to quality, or you were affected by pitfall 3, 7, 8, or 9. **Return to I.**

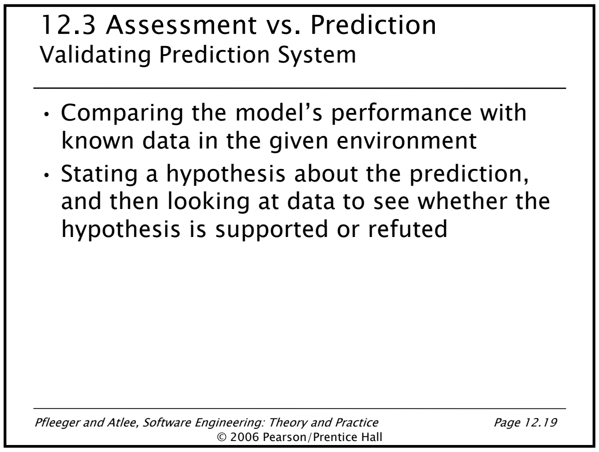
O. YOU WIN! You demonstrate a link between factor X and quality. Report back to your team.

12.2 Selecting An Evaluation Technique
Common Pitfalls in Investigation

Pitfall	Description
1. Confounding	Another factor is causing the effect
2. Cause or effect?	The factor could be a result, not a cause, of the treatment
3. Chance	There is always a small possibility that your result happened by chance
4. Homogeneity	You can find no link because all subjects had the same level of the factor
5. Misclassification	You can find no link because you can not accurately classify each subject's level of the factor
6. Bias	Selection procedures or administration of the study inadvertently bias the result
7. Too short	The short-term effects are different from the long-term ones
8. Wrong amount	The factor would have had an effect, but not in the amount used in the study
9. Wrong situation	The factor has the desired effect, but not in the situation studied

12.3 Assessment vs. Prediction

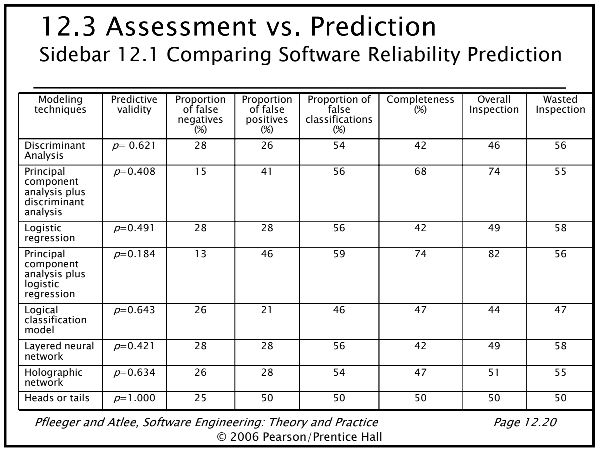
- Assessment system examines an existing entity by characterizing it numerically
- Prediction system predicts characteristic of a future entity; involves a model with associated prediction procedures
  - deterministic prediction (we always get the same output for an input)
  - stochastic prediction (output varies probabilistically)



12.3 Assessment vs. Prediction

Validating Prediction System

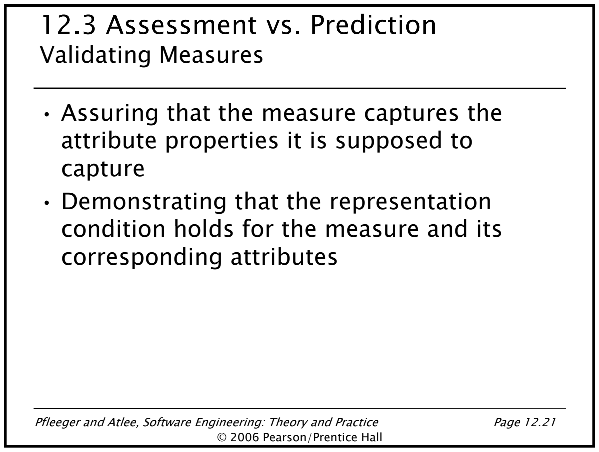
- Comparing the model’s performance with known data in the given environment
- Stating a hypothesis about the prediction, and then looking at data to see whether the hypothesis is supported or refuted



12.3 Assessment vs. Prediction

Sidebar 12.1 Comparing Software Reliability Prediction

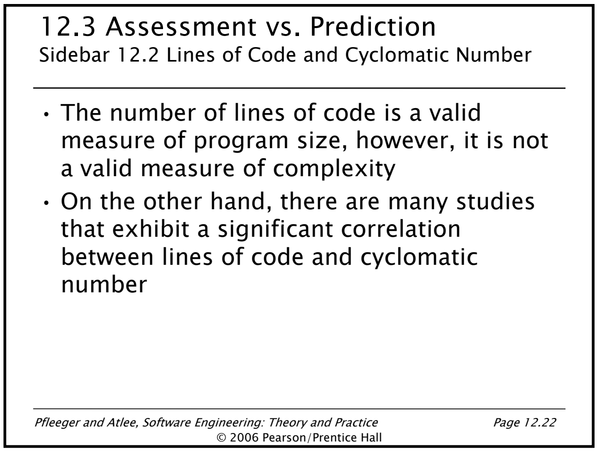
Modeling techniques	Predictive validity	Proportion of false negatives (%)	Proportion of false positives (%)	Proportion of false classifications (%)	Completeness (%)	Overall Inspection	Wasted Inspection
Discriminant Analysis	p= 0.621	28	26	54	42	46	56
Principal component analysis plus discriminant analysis	p=0.408	15	41	56	68	74	55
Logistic regression	p=0.491	28	28	56	42	49	58
Principal component analysis plus logistic regression	p=0.184	13	46	59	74	82	56
Logical classification model	p=0.643	26	21	46	47	44	47
Layered neural network	p=0.421	28	28	56	42	49	58
Holographic network	p=0.634	26	28	54	47	51	55
Heads or tails	p=1.000	25	50	50	50	50	50



12.3 Assessment vs. Prediction

Validating Measures

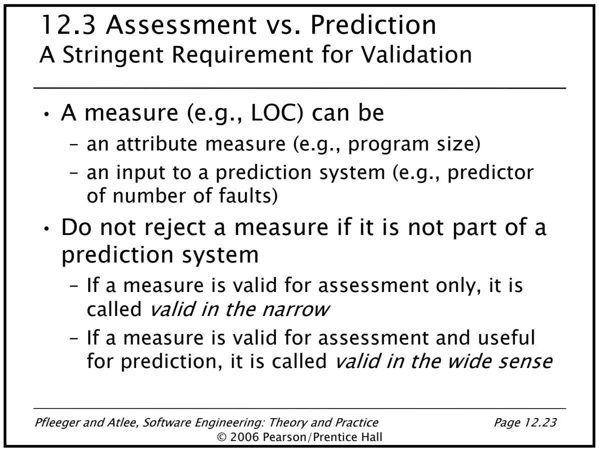
- Assuring that the measure captures the attribute properties it is supposed to capture
- Demonstrating that the representation condition holds for the measure and its corresponding attributes



12.3 Assessment vs. Prediction

Sidebar 12.2 Lines of Code and Cyclomatic Number

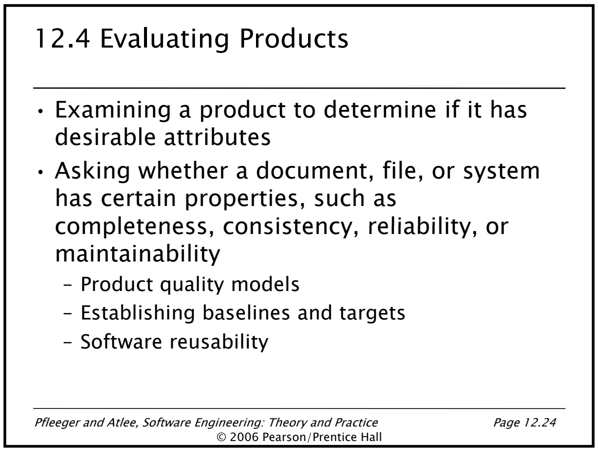
- The number of lines of code is a valid measure of program size, however, it is not a valid measure of complexity
- On the other hand, there are many studies that exhibit a significant correlation between lines of code and cyclomatic number



12.3 Assessment vs. Prediction

A Stringent Requirement for Validation

- A measure (e.g., LOC) can be
  - an attribute measure (e.g., program size)
  - an input to a prediction system (e.g., predictor of number of faults)
- Do not reject a measure if it is not part of a prediction system
  - If a measure is valid for assessment only, it is called *valid in the narrow*
  - If a measure is valid for assessment and useful for prediction, it is called *valid in the wide sense*



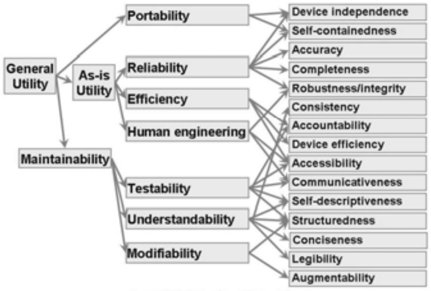
12.4 Evaluating Products

- Examining a product to determine if it has desirable attributes
- Asking whether a document, file, or system has certain properties, such as completeness, consistency, reliability, or maintainability
  - Product quality models
  - Establishing baselines and targets
  - Software reusability

12.4 Evaluating Products
Product Quality Models

- Boehm’s model
- ISO 9126
- Dromey’s Model

12.4 Evaluating Products
Boehm’s Quality Model



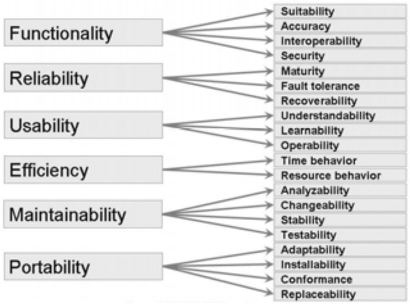
12.4 Evaluating Products
Boehm’s Quality Model (continued)

- Reflects an understanding of quality where the software
  - does what the user wants it do
  - uses computer resources correctly and efficiently
  - is easy for the user to learn and use
  - is well-designed, well-coded, and easily tested and maintained

12.4 Evaluating Products
ISO 9126 Quality Model

- A hierarchical model with six major attributes contributing to quality
  - Each right-hand characteristic is related only to exactly one left-hand attribute

12.4 Evaluating Products
ISO 9126 Quality Model (continued)



12.4 Evaluating Products
ISO 9126 Quality Characteristics

Quality Characteristic	Definition
Functionality	A set of attributes that bears on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs
Reliability	A set of attributes that bears on the capability of software to maintain its performance level under stated conditions for a stated period of time
Usability	A set of attributes that bears on the effort needed for use and on the individual assessment of such use by a stated or implied set of users
Efficiency	A set of attributes that bears on the relationship between the software performance and the amount of resources used under stated conditions
Maintainability	A set of attributes that bears on the effort needed to make specified modifications (which may include corrections, improvements, or adaptations of software to environmental changes and changes in the requirements and functional specifications)
Portability	A set of attributes that bears on the ability of software to be transferred from one environment to another (including the organizational, hardware, or software environment)

12.4 Evaluating Products
Dromey Quality Model

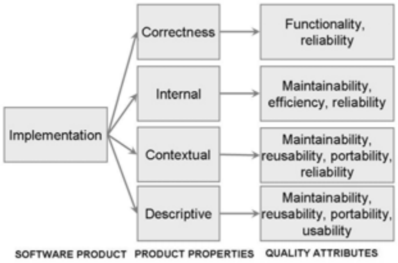
- Product quality depends on the tangible properties of components and component composition
  - Correctness properties
  - Internal properties
  - Contextual properties
  - Descriptive properties

12.4 Evaluating Products
Dromey Quality Model Attributes

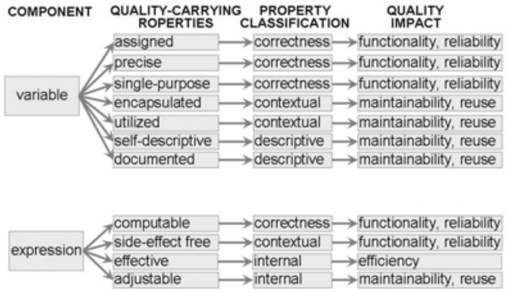
- The six attributes of ISO 9126
- Attributes of reusability
  - machine independence
  - separability
  - configurability
- Process maturity attributes
  - client orientation
  - well-definedness
  - assurance
  - effectiveness

12.4 Evaluating Products
Dromey Quality Model Framework

- Linking product properties to quality attributes


Implementation
Correctness
Internal
Contextual
Descriptive
Functionality, reliability
Maintainability, efficiency, reliability
Maintainability, reusability, portability, reliability
Maintainability, reusability, portability, usability
SOFTWARE PRODUCT
PRODUCT PROPERTIES
QUALITY ATTRIBUTES

12.4 Evaluating Products
Dromey Quality Model Example



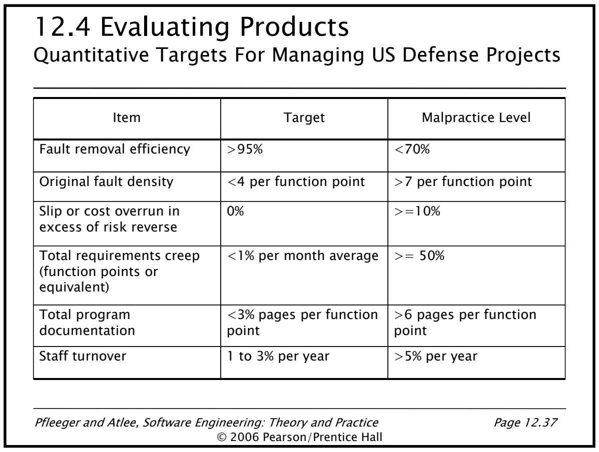
COMPONENT	QUALITY-CARRYING ROPERTIES	PROPERTY CLASSIFICATION	QUALITY IMPACT
variable	assigned	correctness	functionality, reliability
	precise	correctness	functionality, reliability
	single-purpose	correctness	functionality, reliability
	encapsulated	contextual	maintainability, reuse
	utilized	contextual	maintainability, reuse
	self-descriptive	descriptive	maintainability, reuse
	documented	descriptive	maintainability, reuse
expression	computable	correctness	functionality, reliability
	side-effect free	contextual	functionality, reliability
	effective	internal	efficiency
	adjustable	internal	maintainability, reuse

12.4 Evaluating Products
Dromey Quality Model Example (continued)

- The model is based on the following five steps
  - identify a set of high-level quality attributes
  - identify product components
  - identify and classify the most significant, tangible, quality-carrying properties for each component
  - propose a set of axioms for linking product properties to quality attributes
  - evaluate the model, identify its weaknesses, and refine or recreate it

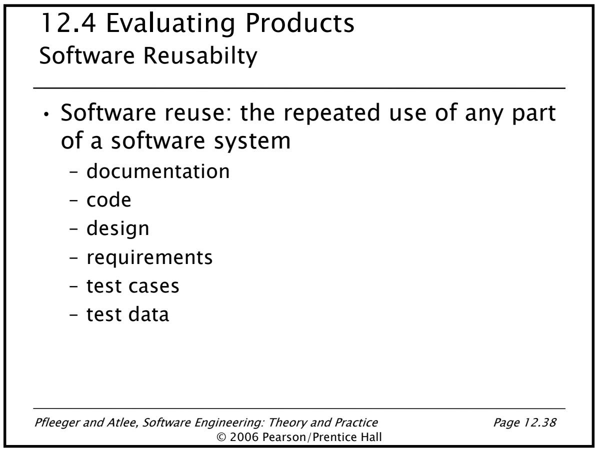
12.4 Evaluating Products
Establishing Baseline and Targets

- A baseline describes the usual or typical result in an organization or category
- Baselines are useful for managing expectations
- A target is a variation of a baseline
  - minimal acceptable behavior



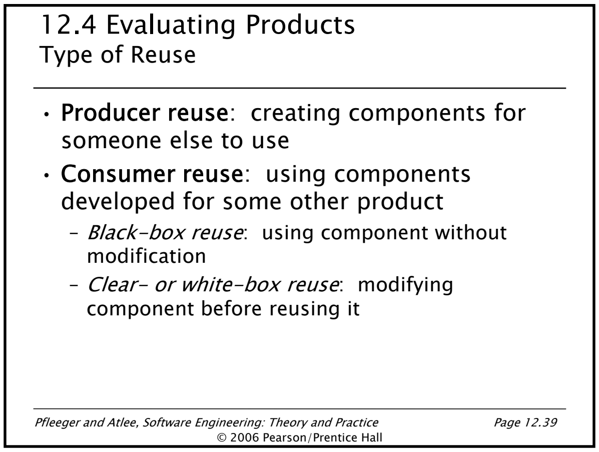
12.4 Evaluating Products
Quantitative Targets For Managing US Defense Projects

Item	Target	Malpractice Level
Fault removal efficiency	>95%	<70%
Original fault density	<4 per function point	>7 per function point
Slip or cost overrun in excess of risk reverse	0%	>=10%
Total requirements creep (function points or equivalent)	<1% per month average	>= 50%
Total program documentation	<3% pages per function point	>6 pages per function point
Staff turnover	1 to 3% per year	>5% per year



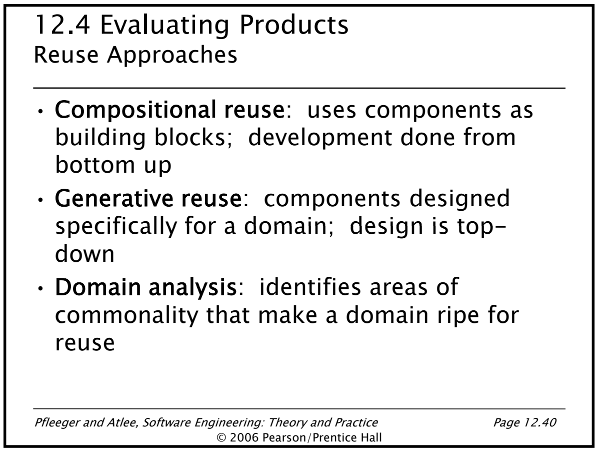
12.4 Evaluating Products
Software Reusabilty

- Software reuse: the repeated use of any part of a software system
  - documentation
  - code
  - design
  - requirements
  - test cases
  - test data



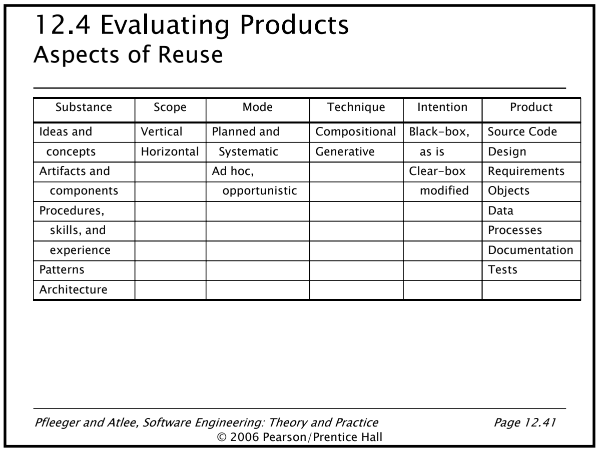
12.4 Evaluating Products
Type of Reuse

- **Producer reuse**: creating components for someone else to use
- **Consumer reuse**: using components developed for some other product
  - *Black-box reuse*: using component without modification
  - *Clear- or white-box reuse*: modifying component before reusing it



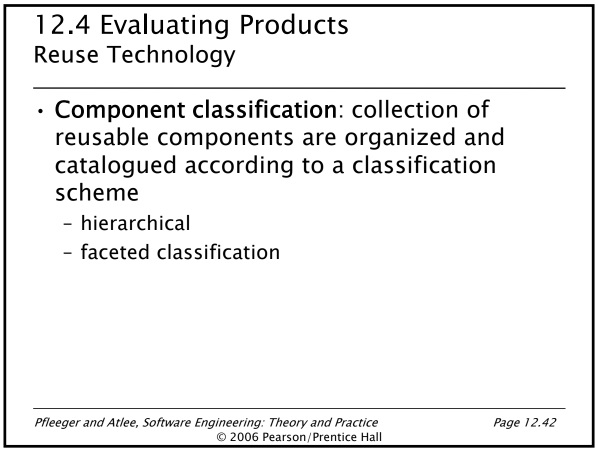
12.4 Evaluating Products
Reuse Approaches

- **Compositional reuse**: uses components as building blocks; development done from bottom up
- **Generative reuse**: components designed specifically for a domain; design is top-down
- **Domain analysis**: identifies areas of commonality that make a domain ripe for reuse



12.4 Evaluating Products
Aspects of Reuse

Substance	Scope	Mode	Technique	Intention	Product
Ideas and	Vertical	Planned and	Compositional	Black-box,	Source Code
concepts	Horizontal	Systematic	Generative	as is	Design
Artifacts and		Ad hoc,		Clear-box	Requirements
components		opportunistic		modified	Objects
Procedures,					Data
skills, and					Processes
experience					Documentation
Patterns					Tests
Architecture					

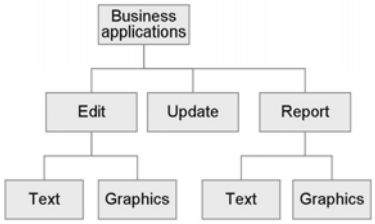


12.4 Evaluating Products
Reuse Technology

- **Component classification**: collection of reusable components are organized and catalogued according to a classification scheme
  - hierarchical
  - faceted classification

12.4 Evaluating Products
Example of A Hierarchical Scheme

- New topic can be added easily at the lowest level



12.4 Evaluating Products
Faceted Classification Scheme

- A facet is a kind of descriptor that helps to identify the component
- Example of the facets of reusable code
  - a application area
  - a function
  - an object
  - a programming language
  - an operating system

12.4 Evaluating Products
Component Retrieval

- A **retrieval system** or **repository**: an automated library that can search for and retrieve a component according to the user's description
- A repository should address a problem of conceptual closeness (values that are similar to but not exactly the same as the desired component)
- Retrieval system can
  - record information about user requests
  - retain descriptive information about the component

12.4 Evaluating Products
Sidebar 12.3 Measuring Reusability

- The measures must
  - address a goal
  - reflect perspective of the person asking the question
- Even if we had a good list of measurements, still it is difficult to determine the characteristic of the most reused component
  - Look at past history
  - Engineering judgment
  - Automated repository

12.4 Evaluating Products
Experience with Reuse

- Raytheon
  - A new system contained an average of 60% reused code increasing productivity by 50%
- GTE Data Services
  - Established incentives and rewards for program authors whenever their components were reused
  - 14% reuse on its project, valued at a savings of $1.5 million
- Nippon Novel
  - Paid 5 cents per line of code to a developer who reused a component

12.4 Evaluating Products
Sidebar 12.4 Software Reuse at Japan's Mainframe Makers

- NEC: reuse library was established to classify, catalog, and document
- Hitachi: integrated software environment, called Eagle, to allow software engineers to reuse standard program patterns and functional procedures
- Fujitsu: created Information Support Center (ISC), that is a regular library staffed with system analysts, software engineers, reuse experts, and switching system domain experts

12.4 Evaluating Products
Benefits of Reuse

- Reuse increases productivity and quality
- Reusing component may increase performance and reliability
- A long term benefit is improved system interoperability

12.4 Evaluating Products
Example of Reuse Success

- Quality, productivity, and time to market at HP

Project Characteristics	HP Project 1	HP Project 2
Size	1100 noncommented source statements	700 noncommented source statements
Quality	51% fault reduction	24% fault reduction
Productivity	57% increase	40% increase
Time to market	Data not available	42% reduction

12.4 Evaluating Products
Example of Cost of Reuse

- Cost to produce and reuse at HP

	Air traffic control system (%)	Menu- and forms Management system (%)	Graphics Firmware (%)
Relative cost to create Reusable code	200	120 to 480	111
Relative cost to reuse	10 to 20	10 to 63	19

12.4 Evaluating Products
Sidebar 12.5 Critical Reuse Success Factors at NTT

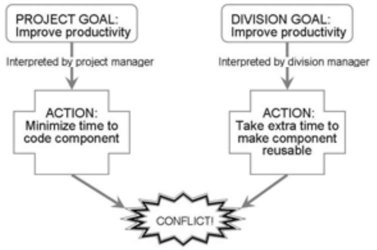
- Success factors at NTT in implementing reuse
  - senior management commitment
  - selecting appropriate target domains
  - systematic development of reusable modules based on domain analysis
  - investing several years of continuous effort in reuse

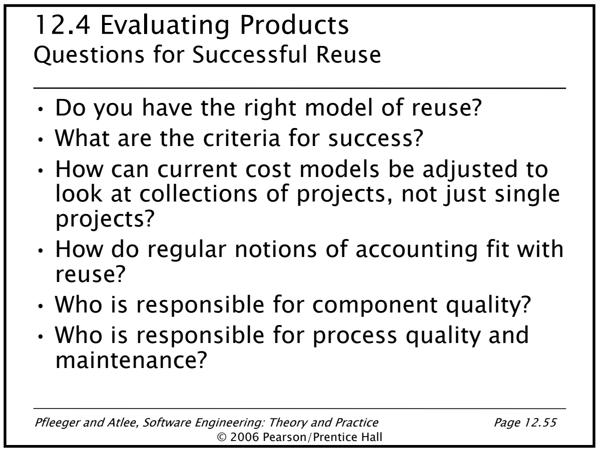
12.4 Evaluating Products
Reuse Lessons

- Reuse goals should be measurable
- Management should resolve reuse goals early
- Different perspectives may generate different questions about reuse
- Every organization must decide at what level to answer reuse questions
- Integrate the reuse process into the development process
- Let your business goals suggest what to measure

12.4 Evaluating Products
Conflicting Interpretation of Goals

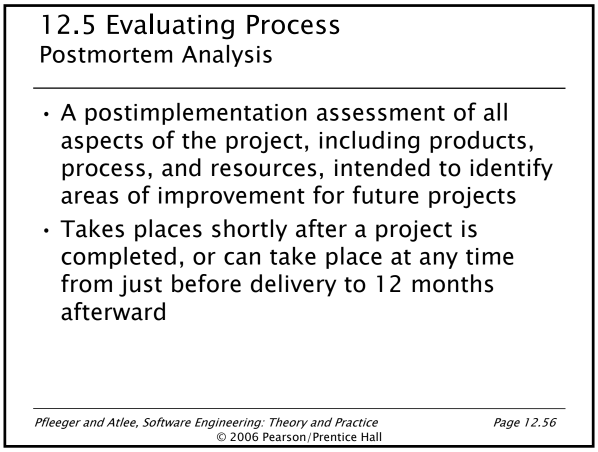
- A division manager's reuse goal may conflict with a project manager's goal, so no reuse ever gets done


PROJECT GOAL:
Improve productivity
Interpreted by project manager
ACTION:
Minimize time to code component
DIVISION GOAL:
Improve productivity
Interpreted by division manager
ACTION:
Take extra time to make component reusable
CONFLICT!



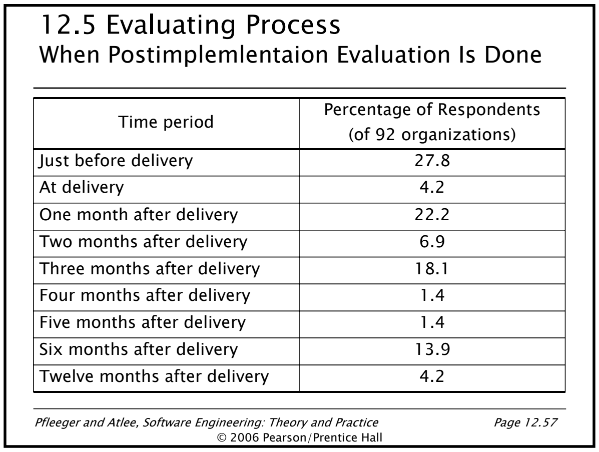
12.4 Evaluating Products
Questions for Successful Reuse

- Do you have the right model of reuse?
- What are the criteria for success?
- How can current cost models be adjusted to look at collections of projects, not just single projects?
- How do regular notions of accounting fit with reuse?
- Who is responsible for component quality?
- Who is responsible for process quality and maintenance?



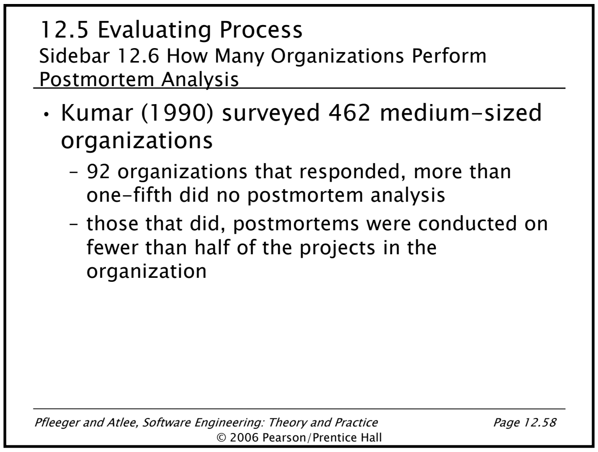
12.5 Evaluating Process
Postmortem Analysis

- A postimplementation assessment of all aspects of the project, including products, process, and resources, intended to identify areas of improvement for future projects
- Takes places shortly after a project is completed, or can take place at any time from just before delivery to 12 months afterward



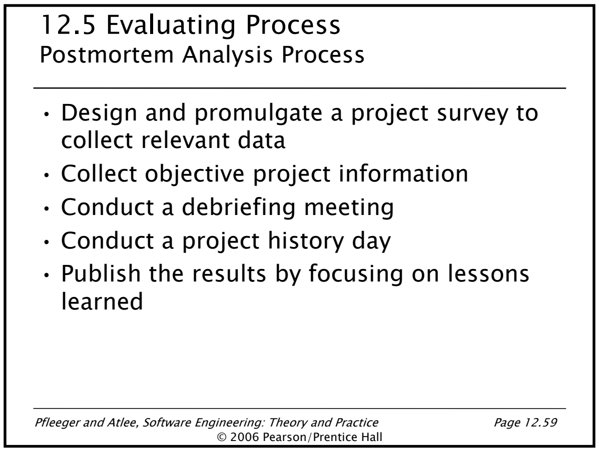
12.5 Evaluating Process
When Postimplemlentaion Evaluation Is Done

Time period	Percentage of Respondents (of 92 organizations)
Just before delivery	27.8
At delivery	4.2
One month after delivery	22.2
Two months after delivery	6.9
Three months after delivery	18.1
Four months after delivery	1.4
Five months after delivery	1.4
Six months after delivery	13.9
Twelve months after delivery	4.2



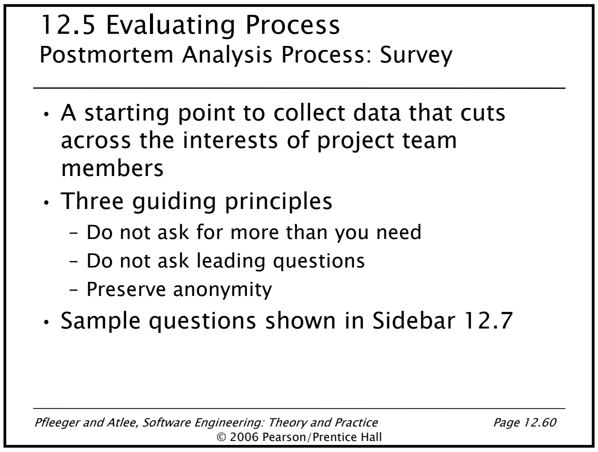
12.5 Evaluating Process
Sidebar 12.6 How Many Organizations Perform Postmortem Analysis

- Kumar (1990) surveyed 462 medium-sized organizations
  - 92 organizations that responded, more than one-fifth did no postmortem analysis
  - those that did, postmortems were conducted on fewer than half of the projects in the organization



12.5 Evaluating Process
Postmortem Analysis Process

- Design and promulgate a project survey to collect relevant data
- Collect objective project information
- Conduct a debriefing meeting
- Conduct a project history day
- Publish the results by focusing on lessons learned



12.5 Evaluating Process
Postmortem Analysis Process: Survey

- A starting point to collect data that cuts across the interests of project team members
- Three guiding principles
  - Do not ask for more than you need
  - Do not ask leading questions
  - Preserve anonymity
- Sample questions shown in Sidebar 12.7

12.5 Evaluating Process

Sidebar 12.7 Sample Survey Questions From Wildfire Survey

- Were interdivisional lines of responsibility clearly defined throughout the project?
- Did project-related meetings make effective use of your time?
- Were you empowered to participate in discussion regarding issues that affected your work?
- Did schedule changes and related decisions involve the right people?
- Was project definition done by the appropriate individuals?
- Was the build process effective for the component area you worked on?
- What is your primary function on this project?

12.5 Evaluating Process

Postmortem Analysis Process: Objective Information

- Obtain objective information to complement the survey opinions
- Collier, Demarco, and Fearey suggest three kinds of measurements: cost, schedule, and quality
  - Cost measurements might include
    - person-months of effort
    - total lines of code
    - number of lines of code changed or added
    - number of interfaces

12.5 Evaluating Process

Postmortem Analysis Process: Debriefing Meeting

- Allows team members to report what did and did not go well on the project
- Project leader can probe more deeply to identify the root cause of positive and negative effects
- Some team members may raise issues not covered in the survey questions
- Debriefing meetings should be loosely structured

12.5 Evaluating Process

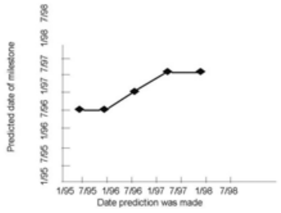
Postmortem Analysis Process: Project History Day

- Objective: identify the root causes of the key problems
- Involves a limited number of participants who know something about key problems
- Review schedule predictability charts
  - Show where problems occurred
  - Spark discussion about possible causes of each problem

12.5 Evaluating Process

Postmortem Analysis Process: Schedule-Predictability Charts

- For each key project milestone, the chart shows when the predictions was made compared with the completion date



12.5 Evaluating Process

Postmortem Analysis Process: Publish Results

- Objective: Share results with the project team
- Participants in the project history day write a letter to managers, peers, developers
- The letter has four parts
  - Introduction to the project
  - A summary of postmortem's positive findings
  - A summary of three worst factors that kept the team from meeting its goals
  - Suggestions for improvement activities

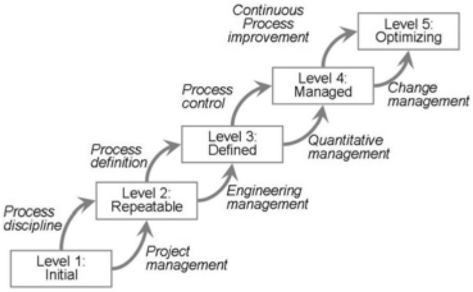
12.5 Evaluating Process
Process Maturity Models

- Capability Maturity Model (CMM)
- Software Process Improvement and Capability dEtermination (SPICE)
- ISO 9000

12.5 Evaluating Process
Capability Maturity Model

- Developed by Software Engineering Institute
- There are five levels of maturity
- Each level is associated with a set of key process areas

12.5 Evaluating Process
CMM Levels of Maturity



12.5 Evaluating Process
CMM Maturity Levels

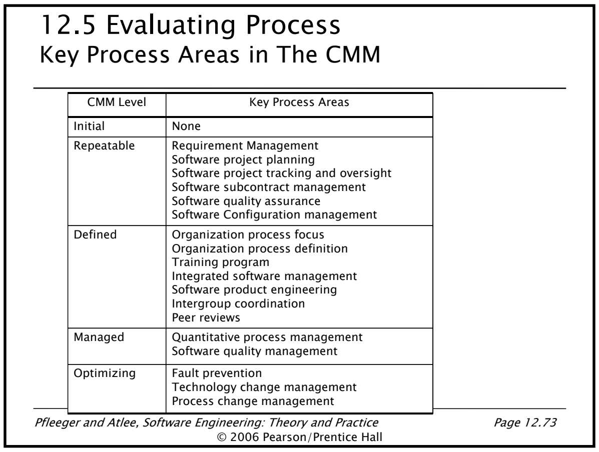
- Level 1: Initial
- Level 2: Repeatable
- Level 3: Defined
- Level 4: Managed
- Level 5: Optimizing

12.5 Evaluating Process
CMM Level 1

- Initial: describes a software development process that is ad hoc or even chaotic
- It is difficult even to write down or depict the overall process
- No key process areas at this level

12.5 Evaluating Process
Required Questions for Level 1 of The Process Maturity Model

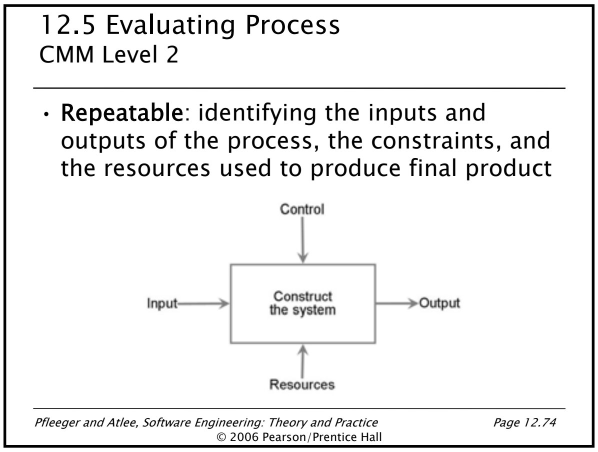
Question number	Question
1.1.3	Does the software quality assurance function have a management reporting channel separate from the software development project management?
1.1.6	Is there a software configuration control function for each project that involves software development?
2.1.3	Is a formal process used in the management review of each software development prior to making contractual commitments?
2.1.14	Is a formal procedure used to make estimates of software size?
2.1.15	Is a formal procedure used to produce software development schedules?
2.1.16	Are formal procedures applied to estimating software development cost?
2.2.2	Are profiles of software size maintained for each software configuration item over time?
2.2.4	Are statistics on software code and test errors gathered?
2.4.1	Does senior management have a mechanism for the regular review of the status of software development projects?
2.4.7	Do software development first-line managers sign off on their schedule and cost estimates?
2.4.9	Is a mechanism used for controlling changes to the software requirements?
2.4.17	Is a mechanism used for controlling changes to the code?



12.5 Evaluating Process
Key Process Areas in The CMM

CMM Level	Key Process Areas
Initial	None
Repeatable	Requirement Management Software project planning Software project tracking and oversight Software subcontract management Software quality assurance Software Configuration management
Defined	Organization process focus Organization process definition Training program Integrated software management Software product engineering Intergroup coordination Peer reviews
Managed	Quantitative process management Software quality management
Optimizing	Fault prevention Technology change management Process change management

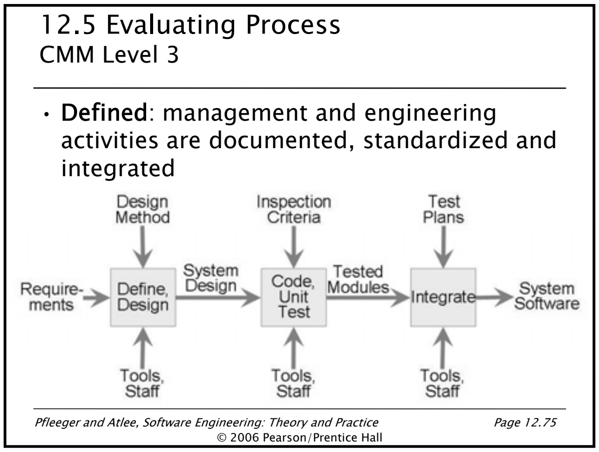
Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.73*
© 2006 Pearson/Prentice Hall



12.5 Evaluating Process
CMM Level 2

- **Repeatable**: identifying the inputs and outputs of the process, the constraints, and the resources used to produce final product

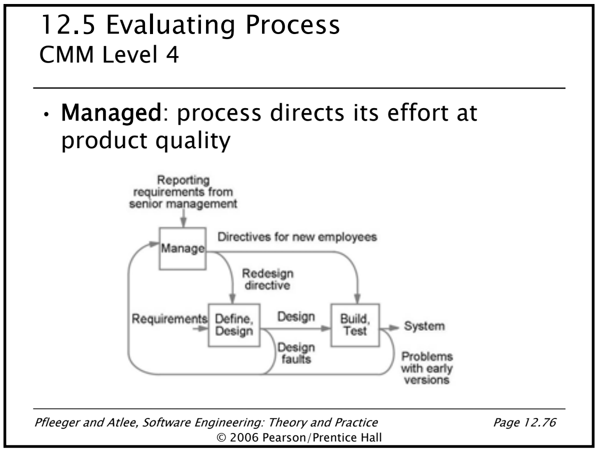
Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.74*
© 2006 Pearson/Prentice Hall



12.5 Evaluating Process
CMM Level 3

- **Defined**: management and engineering activities are documented, standardized and integrated

Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.75*
© 2006 Pearson/Prentice Hall

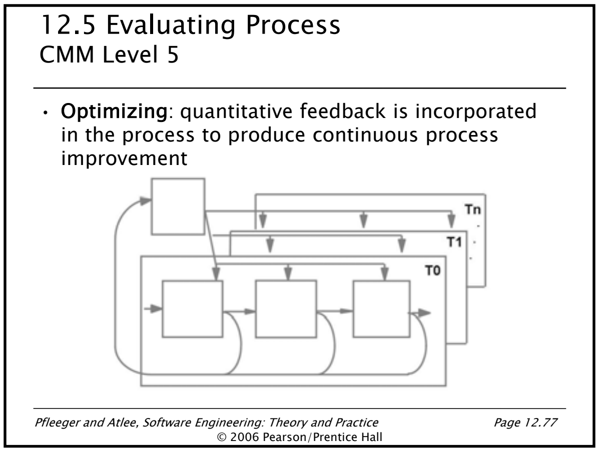


12.5 Evaluating Process
CMM Level 4

- **Managed**: process directs its effort at product quality

Reporting requirements from senior management
Manage
Directives for new employees
Redesign directive
Requirements
Define, Design
Design
Build, Test
System
Design faults
Problems with early versions

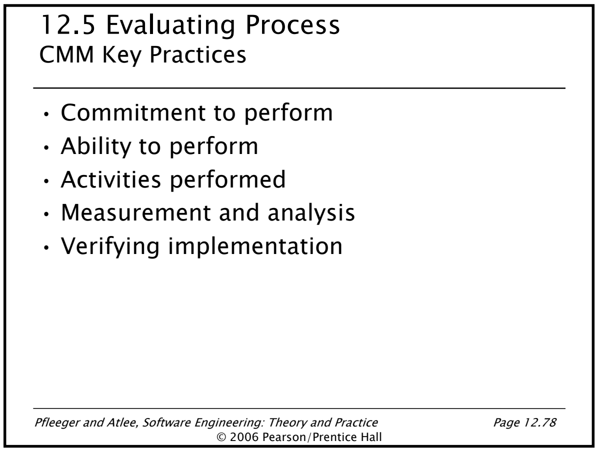
Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.76*
© 2006 Pearson/Prentice Hall



12.5 Evaluating Process
CMM Level 5

- **Optimizing**: quantitative feedback is incorporated in the process to produce continuous process improvement

Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.77*
© 2006 Pearson/Prentice Hall



12.5 Evaluating Process
CMM Key Practices

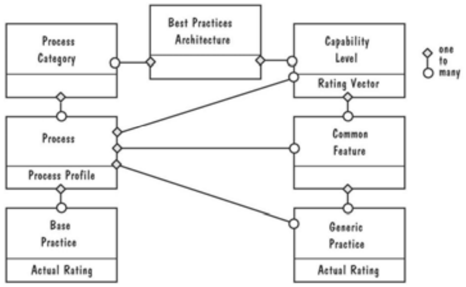
- Commitment to perform
- Ability to perform
- Activities performed
- Measurement and analysis
- Verifying implementation

Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.78*
© 2006 Pearson/Prentice Hall

12.5 Evaluating Process
SPICE

- SPICE is intended to harmonize and extend the existing approaches (e.g., CMM, BOOTSTRAP)
- SPICE is recommended for process improvement and capability determination
- Two types of practices
  - *Base practices*: essential activities of a specific process
  - *Generic practices*: institutionalization (implement a process in a general way)

12.5 Evaluating Process
SPICE Architecture for Process Assessment



12.5 Evaluating Process
SPICE Functional View Activities/Processes

- Customer-supplied: processes that affect the customer directly
- Engineering: processes that specify, implement, or maintain the system
- Project: processes that establish the project, and coordinate and manage resources
- Support: processes that enable other processes
- Organizational: processes that establish business goals

12.5 Evaluating Process
SPICE Six Levels of Capability

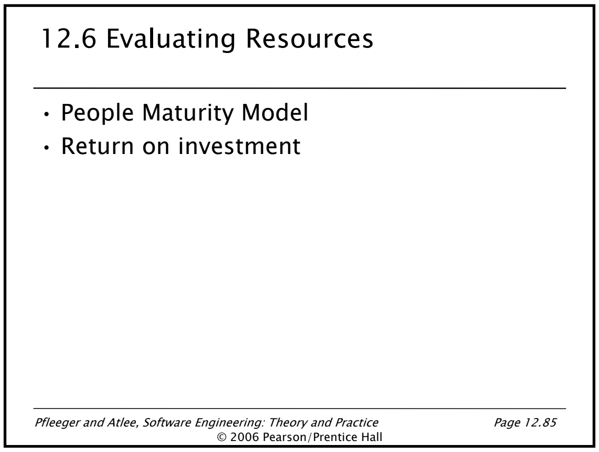
- 0: Not performed – failure to perform
- 1: Performed informally: not planned and tracked
- 2: Planned and tracked: verified according to the specified procedures
- 3: Well-defined: well-defined process using approved processes
- 4: Quantitatively controlled: detailed performance measures
- 5: Continuously improved: quantitative targets for effectiveness and efficiency based on business goals

12.5 Evaluating Process
ISO 9000

- Produced by The International Standards Organization (ISO)
- Standard 9001 is most applicable to the way we develop and maintain software
- Used to regulate internal quality and to ensure the quality suppliers

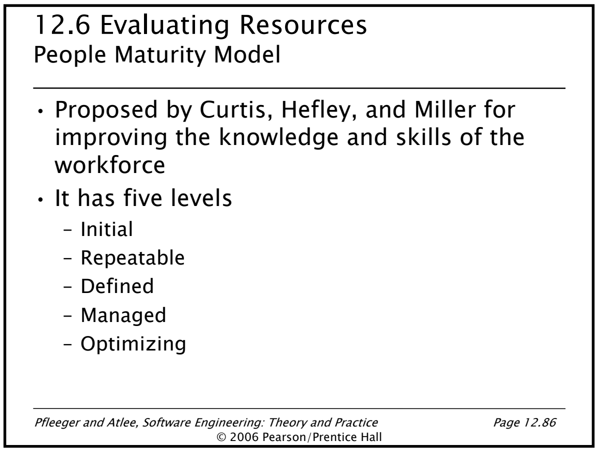
12.5 Evaluating Process
ISO 9001 Clauses

Clause number	Subject matter
4.1	Management responsibility
4.2	Quality system
4.3	Contract review
4.4	Design control
4.5	Document and data control
4.6	Purchasing
4.7	Control of customer-supplied product
4.8	Product identification and traceability
4.9	Process control
4.10	Inspection and testing
4.11	Control of inspection, measuring, and test equipment
4,12	Inspection and test status
4,.13	Control of nonconforming product
4.14	Corrective and preventive action
4.15	Handling, storage, packaging, preservation, and delivery
4.16	Control of quality records
4,17	Internal quality audits
4.18	Training
4.19	Servicing
4.20	Statistical techniques



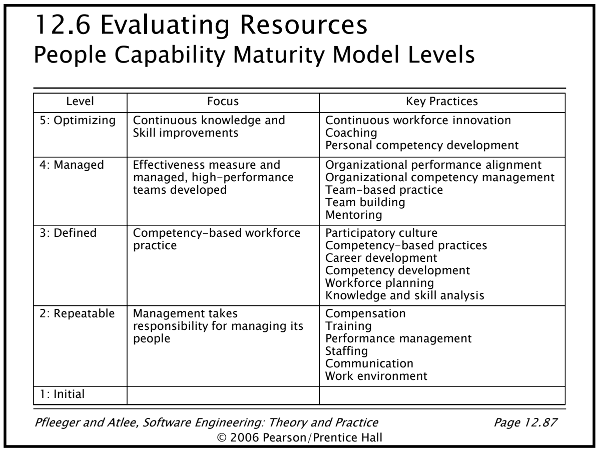
12.6 Evaluating Resources

- People Maturity Model
- Return on investment



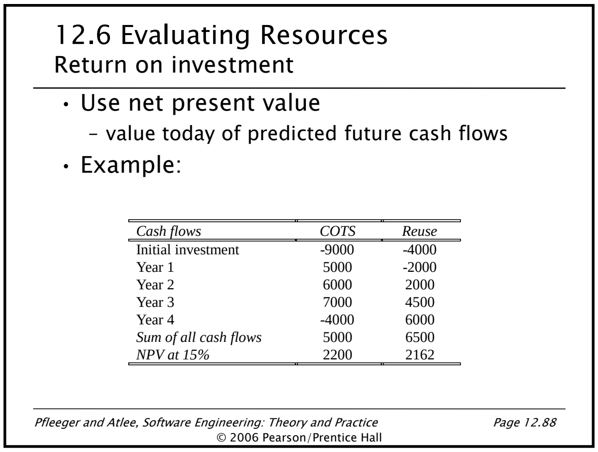
12.6 Evaluating Resources
People Maturity Model

- Proposed by Curtis, Hefley, and Miller for improving the knowledge and skills of the workforce
- It has five levels
  - Initial
  - Repeatable
  - Defined
  - Managed
  - Optimizing



12.6 Evaluating Resources
People Capability Maturity Model Levels

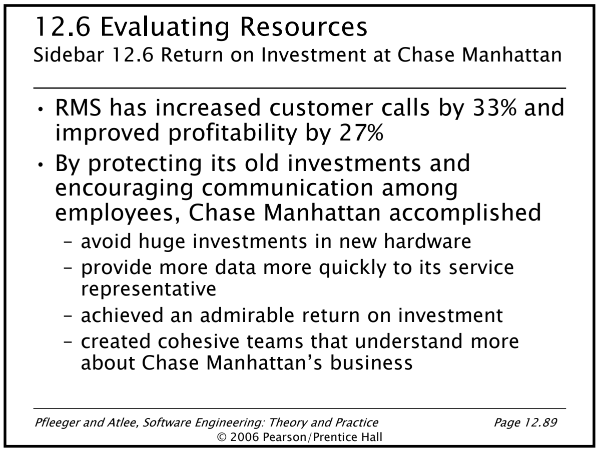
Level	Focus	Key Practices
5: Optimizing	Continuous knowledge and Skill improvements	Continuous workforce innovation Coaching Personal competency development
4: Managed	Effectiveness measure and managed, high-performance teams developed	Organizational performance alignment Organizational competency management Team-based practice Team building Mentoring
3: Defined	Competency-based workforce practice	Participatory culture Competency-based practices Career development Competency development Workforce planning Knowledge and skill analysis
2: Repeatable	Management takes responsibility for managing its people	Compensation Training Performance management Staffing Communication Work environment
1: Initial		



12.6 Evaluating Resources
Return on investment

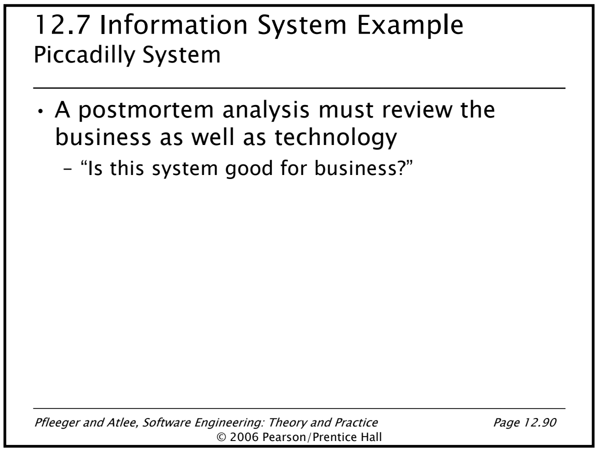
- Use net present value
  - value today of predicted future cash flows
- Example:

Cash flows	*COTS*	*Reuse*
Initial investment	-9000	-4000
Year 1	5000	-2000
Year 2	6000	2000
Year 3	7000	4500
Year 4	-4000	6000
Sum of all cash flows	5000	6500
NPV at 15%	2200	2162



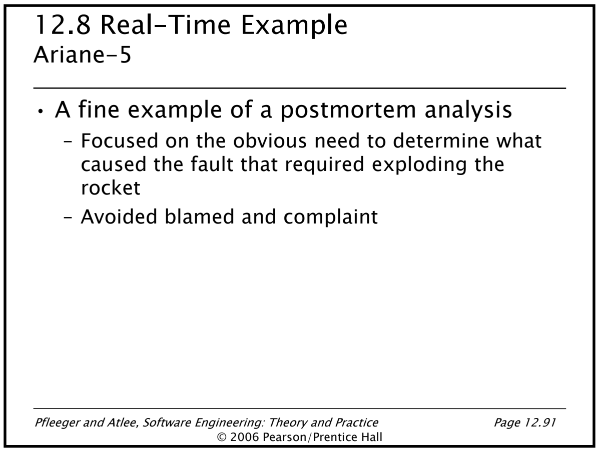
12.6 Evaluating Resources
Sidebar 12.6 Return on Investment at Chase Manhattan

- RMS has increased customer calls by 33% and improved profitability by 27%
- By protecting its old investments and encouraging communication among employees, Chase Manhattan accomplished
  - avoid huge investments in new hardware
  - provide more data more quickly to its service representative
  - achieved an admirable return on investment
  - created cohesive teams that understand more about Chase Manhattan's business



12.7 Information System Example
Piccadilly System

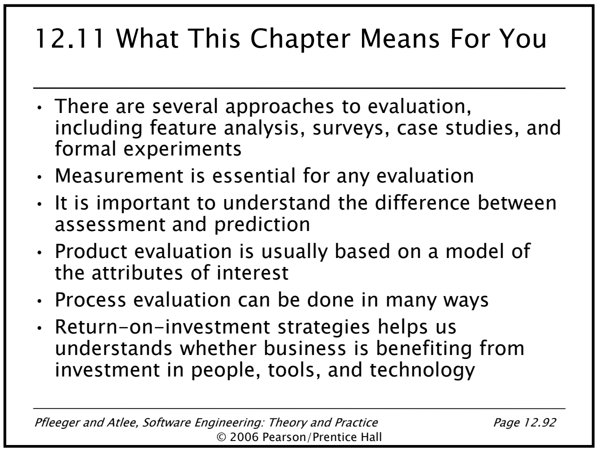
- A postmortem analysis must review the business as well as technology
  - "Is this system good for business?"



12.8 Real-Time Example
Ariane-5

- A fine example of a postmortem analysis
  - Focused on the obvious need to determine what caused the fault that required exploding the rocket
  - Avoided blamed and complaint

Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.91*



12.11 What This Chapter Means For You

- There are several approaches to evaluation, including feature analysis, surveys, case studies, and formal experiments
- Measurement is essential for any evaluation
- It is important to understand the difference between assessment and prediction
- Product evaluation is usually based on a model of the attributes of interest
- Process evaluation can be done in many ways
- Return-on-investment strategies helps us understands whether business is benefiting from investment in people, tools, and technology

Pfleeger and Atlee, Software Engineering: Theory and Practice *Page 12.92*